Open-Source Software Practice

Lab 02. Introduction

Instructor: Jaemin Jo (조재민, jmjo@skku.edu)
Interactive Data Computing Lab (*IDCLab*),
College of Computing and Informatics,
Sungkyunkwan University

Schedule

- The final exam will be offline.

Week	Topic	Week	Topic
1	Introduction + GitHub Tour	9	Web Application
2	Git Basics	10	Desktop Application
3	Git Advanced	11	Collaboration
4	Git Misc. + Code Editor	12	Client and Server Arch.
5	Node and JavaScript	13	Final Exam
6	JavaScript Advanced	14	Project Presentation
7	Testing and Publishing	15	Project Presentation
8	HTML & CSS		

Lab Session Rules

- Tuesday 6 p.m.
- Watch the lecture video before joining the lab session.
  - I will assume that you have watched the video.
- You can use either Korean or English, but the lecture will be in Korean.
- ~ 1 or 1.5 hours
- TA: Jiwon Choi (jasonchoi3@g.skku.edu)

Lab Session Rules

- Offline
  - 85718
  - Bring your laptop.
  - Wear a mask.
  - Use 전자출결시스템 (e-Attendance system).
  - Raise your hand if you have questions. TA will come to you.

- Online
  - Zoom
  - Post your question to the chatbox.
  - Please be concrete. We cannot see your screen.

Environment

- We assume you are using a Windows machine.
  - e.g., installation instructions
  - Windows 10 or 11
  - You can use Google for OS-specific issues.
  - If you cannot address the issues by yourself, raise your hand.
- Boot Camp, Parallels, VMware, VirtualBox, …

Troubleshooting

- If you have a problem during the class,

1. Carefully read the slides again.
2. Google it first **in English**.
3. Check out if the same question has been already answered in the chat box.
4. Raise your hand or post the question.
5. If the problem persists, consult me or TA after the session (or during the break)

- Remember that TAs are limited resources.
- Be nice to them!

Team Building for the Final Project

- 4 random people will be in a team.
- Peer evaluation
- if (# of students) % 4, there will be teams of fewer people.

- I will announce the team list once the class roster is finalized.

Goals

1. Install VS Code
2. Install Git
3. Play with Git locally

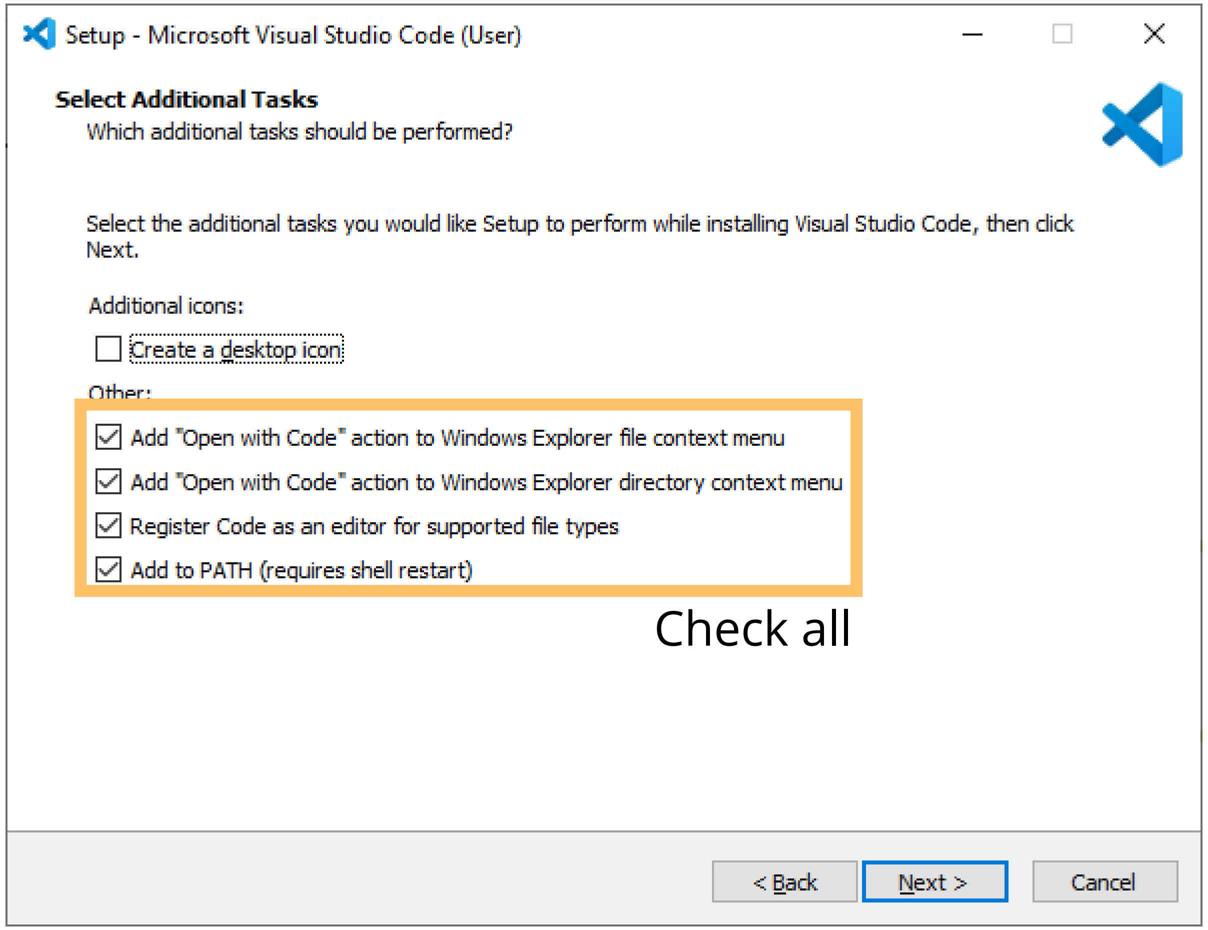
- Why VS Code before Git?
  - We will set it as Git's default text editor.

Install VS Code

- https://code.visualstudio.com/

- You can use any text editor of your preference, but I strongly recommend VS Code.

Install VS Code


Setup - Microsoft Visual Studio Code (User)

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:

- [ ] Create a desktop icon

Other:

- [✓] Add "Open with Code" action to Windows Explorer file context menu
- [✓] Add "Open with Code" action to Windows Explorer directory context menu
- [✓] Register Code as an editor for supported file types
- [✓] Add to PATH (requires shell restart)

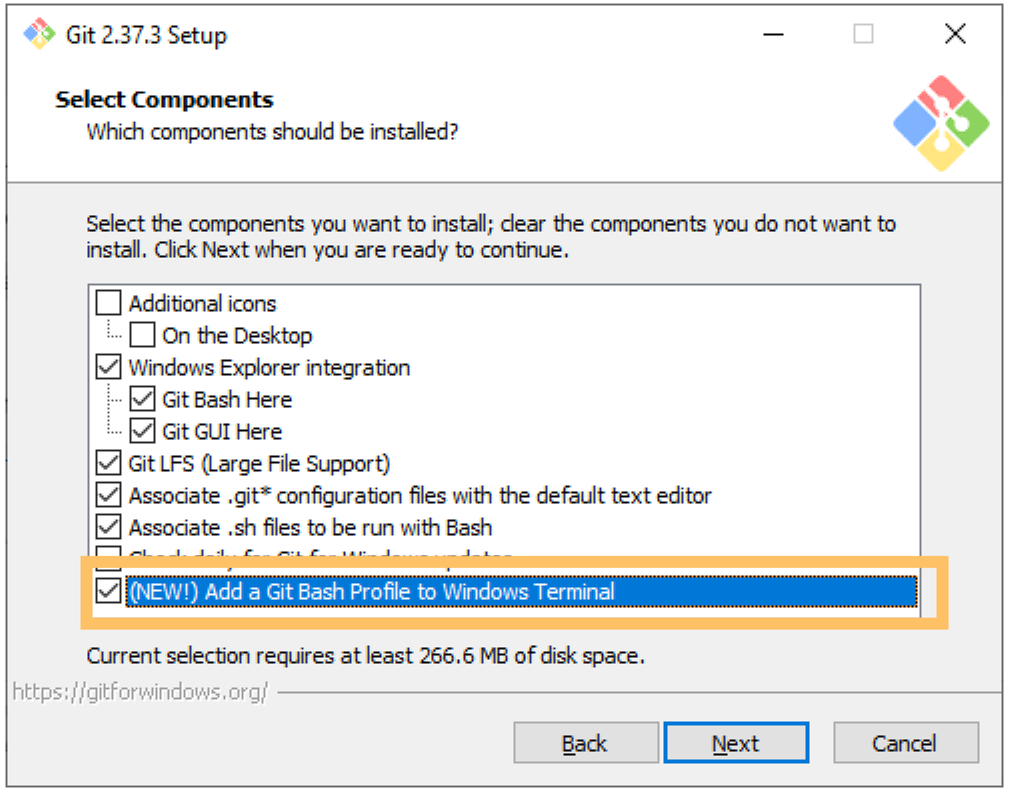
Check all

Install Git

- https://git-scm.com/

- Standalone Installer – 64-bit Git for Windows Setup

Install Git



Git 2.37.3 Setup

Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

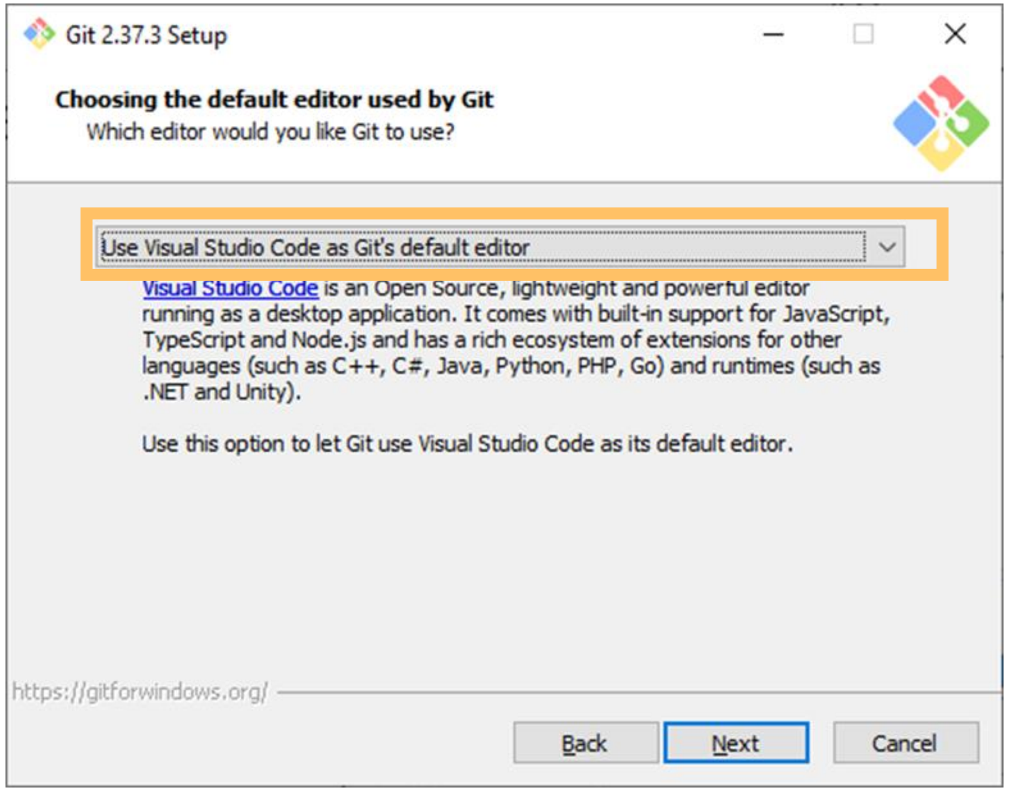
- [ ] Additional icons
  - [ ] On the Desktop
- [x] Windows Explorer integration
  - [x] Git Bash Here
  - [x] Git GUI Here
- [x] Git LFS (Large File Support)
- [x] Associate .git* configuration files with the default text editor
- [x] Associate .sh files to be run with Bash
- [ ] Check daily for Git for Windows updates
- [x] (NEW!) Add a Git Bash Profile to Windows Terminal

Current selection requires at least 266.6 MB of disk space.

https://gitforwindows.org/

Check this option.
It allows you to use Git commands on Windows terminal as well as Git bash.

Install Git


Git 2.37.3 Setup

Choosing the default editor used by Git

Which editor would you like Git to use?

Use Visual Studio Code as Git's default editor

Visual Studio Code is an Open Source, lightweight and powerful editor running as a desktop application. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).

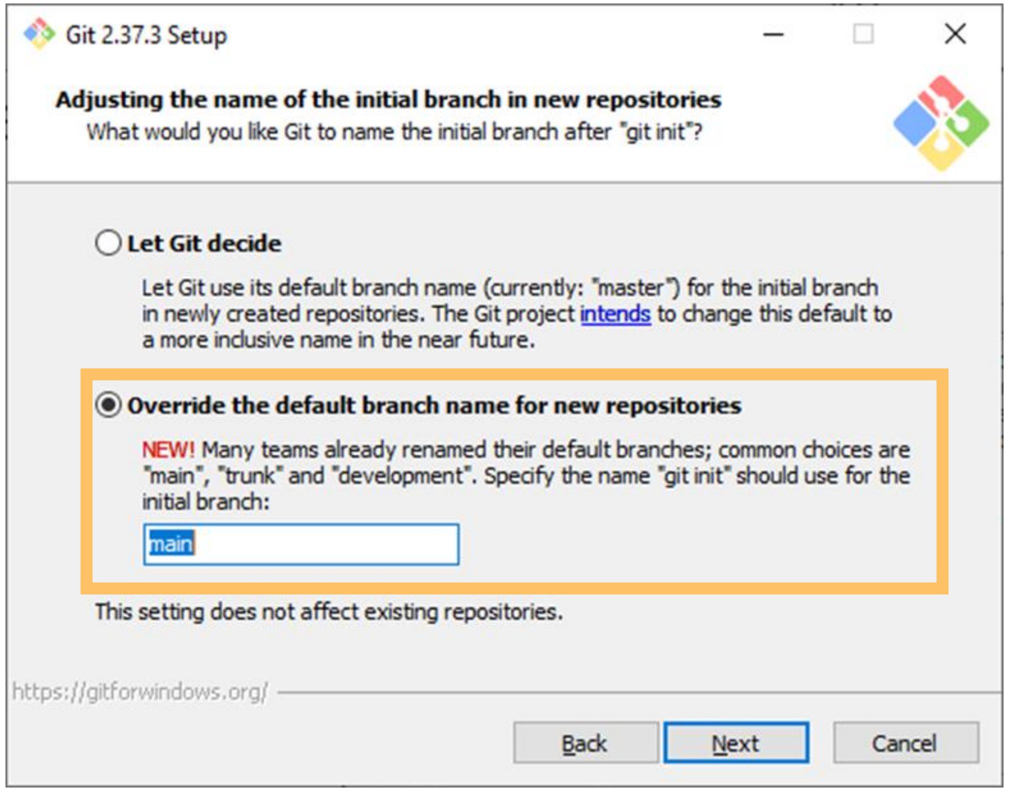
Use this option to let Git use Visual Studio Code as its default editor.

https://gitforwindows.org/

Set VS Code as the default.

If this option is not present, make sure install VSC and relaunch the installer.

Install Git



Git 2.37.3 Setup

Adjusting the name of the initial branch in new repositories

What would you like Git to name the initial branch after "git init"?

○ **Let Git decide**

Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project intends to change this default to a more inclusive name in the near future.

● **Override the default branch name for new repositories**

NEW! Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

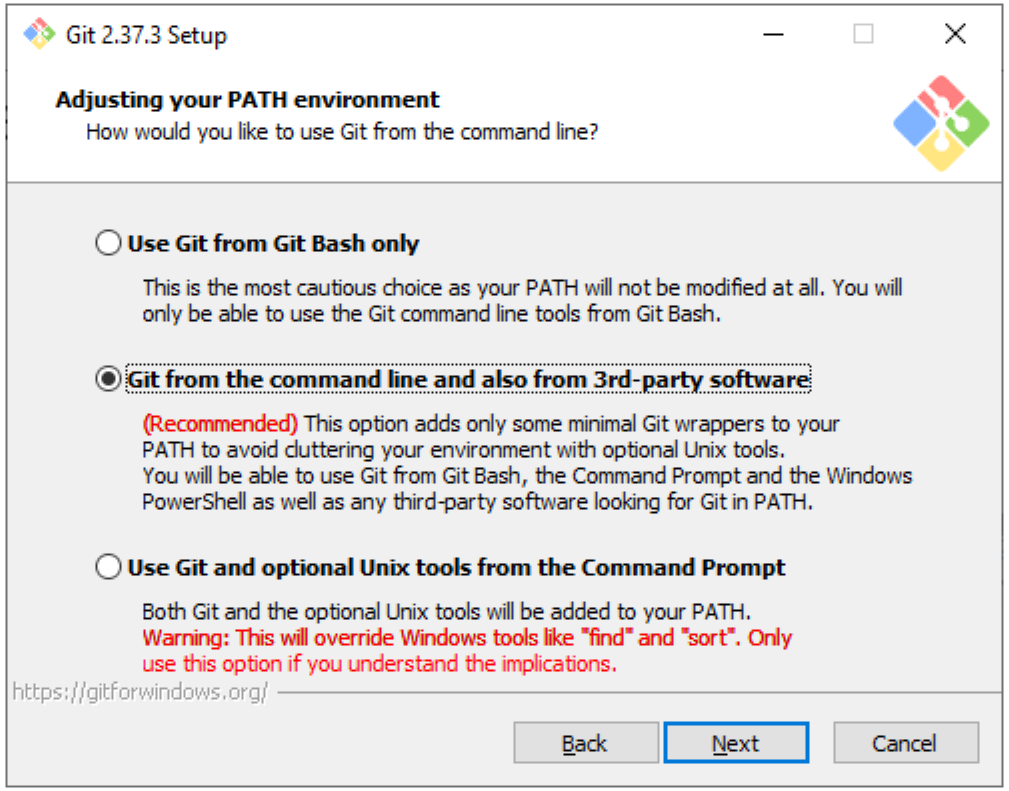
main

This setting does not affect existing repositories.

https://gitforwindows.org/

Let's use *main* as the default branch name following GitHub.

Install Git



Git 2.37.3 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

○ **Use Git from Git Bash only**

This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

[✓ **Git from the command line and also from 3rd-party software**]

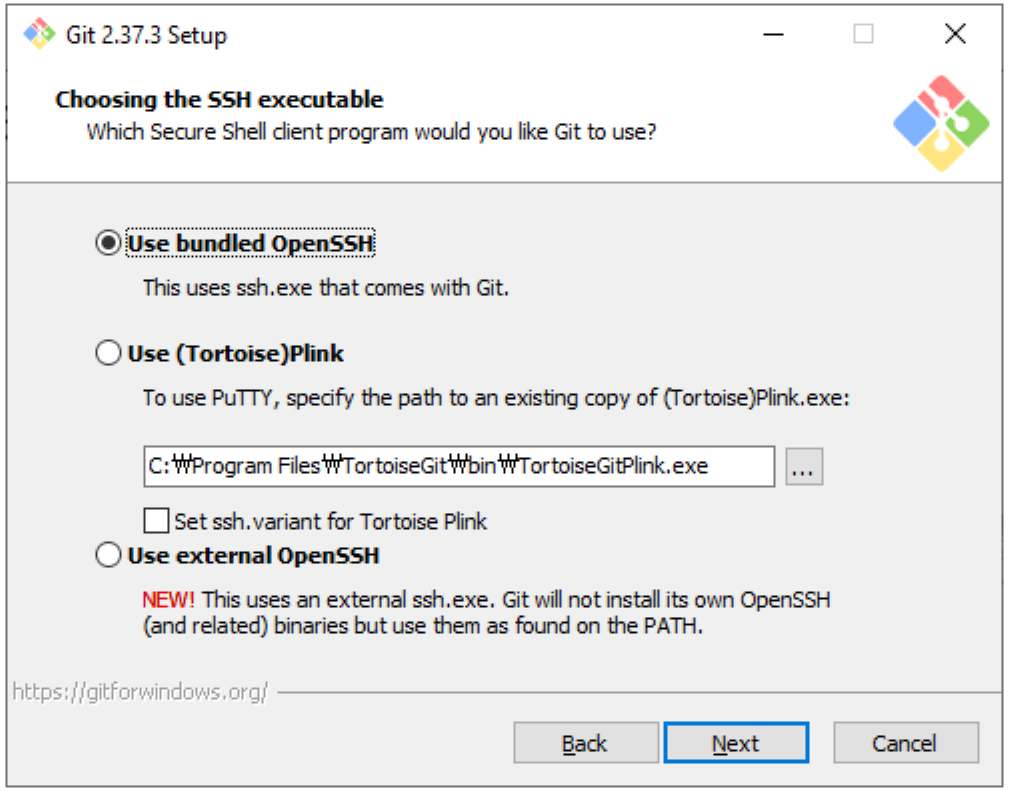
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

○ **Use Git and optional Unix tools from the Command Prompt**

Both Git and the optional Unix tools will be added to your PATH. Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

Use the default.

Install Git


Git 2.37.3 Setup

Choosing the SSH executable
Which Secure Shell client program would you like Git to use?

[✓ **Use bundled OpenSSH**]
This uses ssh.exe that comes with Git.

Use (Tortoise)Plink
To use PuTTY, specify the path to an existing copy of (Tortoise)Plink.exe:

C:₩Program Files₩TortoiseGit₩bin₩TortoiseGitPlink.exe

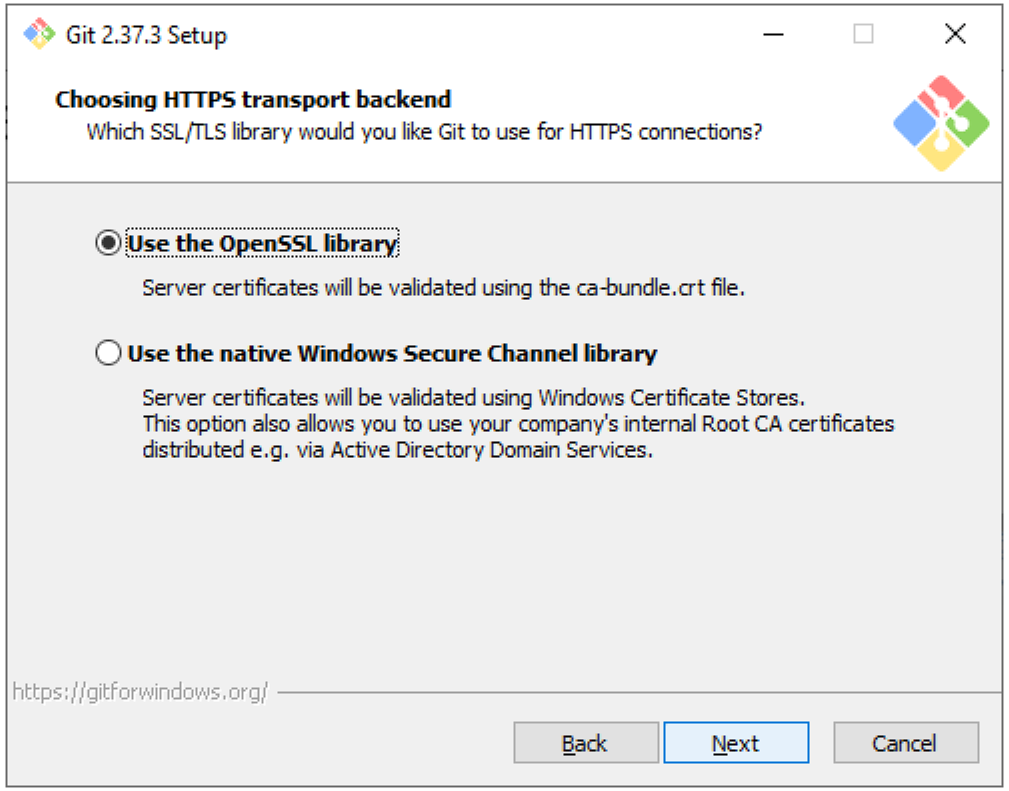
☐ Set ssh.variant for Tortoise Plink

Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.

https://gitforwindows.org/

Use the default.

Install Git



Git 2.37.3 Setup

Choosing HTTPS transport backend

Which SSL/TLS library would you like Git to use for HTTPS connections?

[✓ **Use the OpenSSL library**]

Server certificates will be validated using the ca-bundle.crt file.

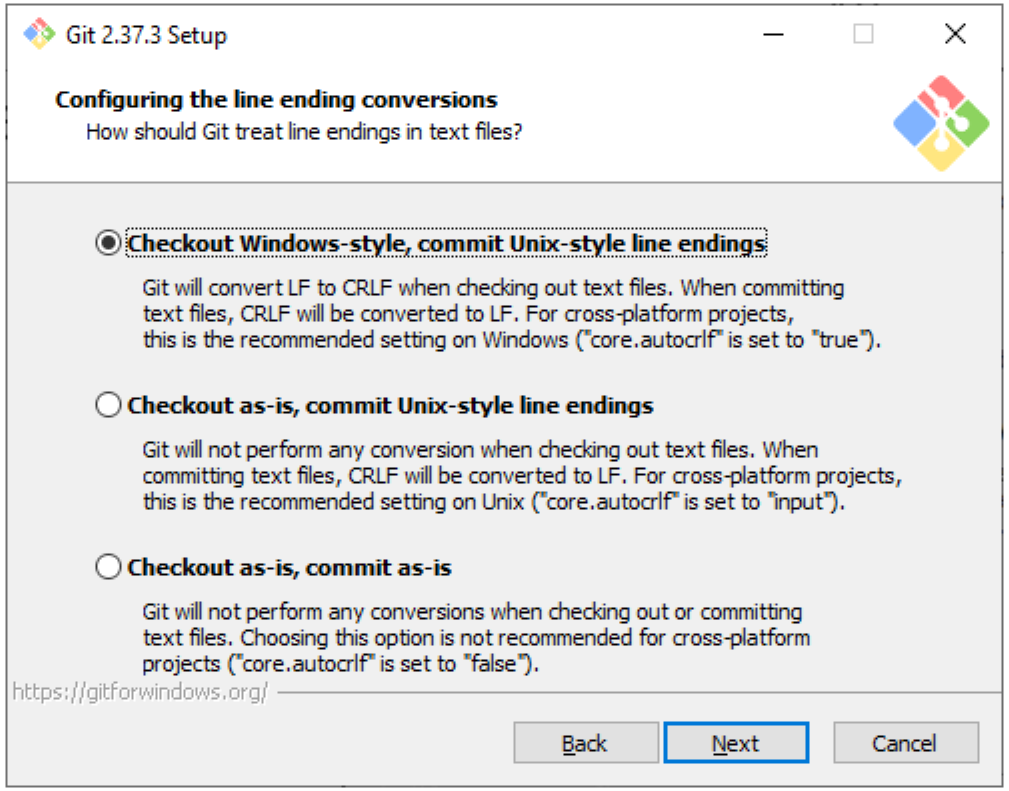
Use the native Windows Secure Channel library

Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.

https://gitforwindows.org/

Use the default.

Install Git



Git 2.37.3 Setup

Configuring the line ending conversions

How should Git treat line endings in text files?

[✓ **Checkout Windows-style, commit Unix-style line endings**]

Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

Checkout as-is, commit Unix-style line endings

Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

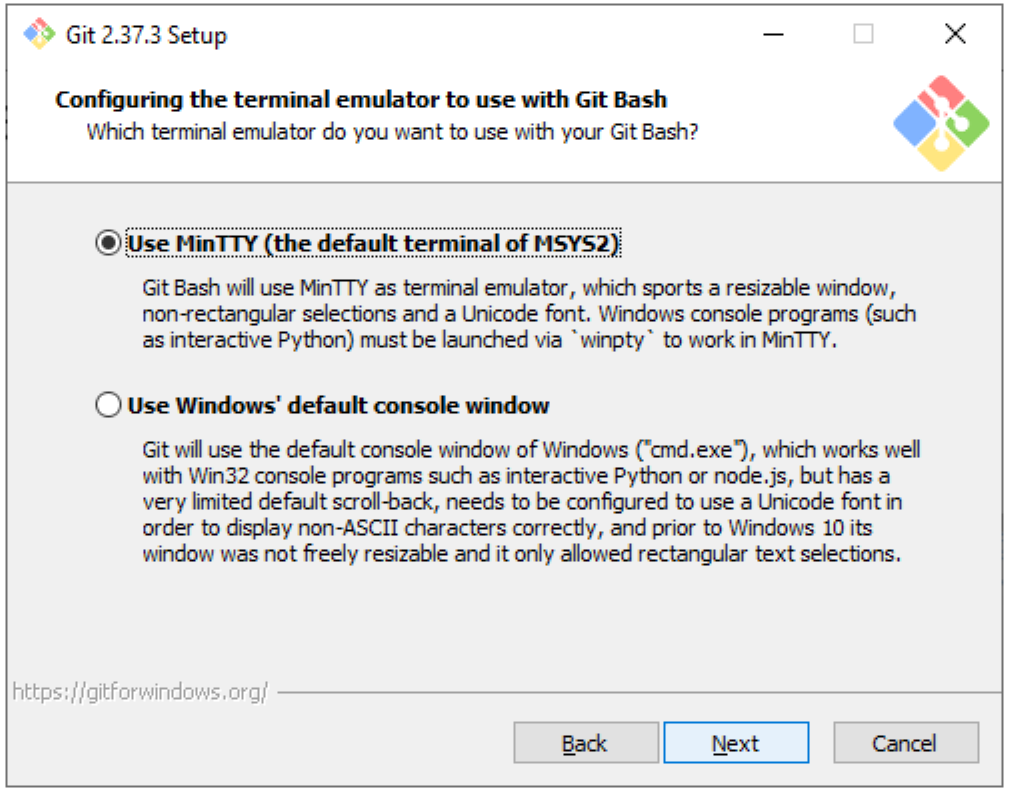
Checkout as-is, commit as-is

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

https://gitforwindows.org/

Use the default.

Install Git



Git 2.37.3 Setup

Configuring the terminal emulator to use with Git Bash

Which terminal emulator do you want to use with your Git Bash?

[✓ **Use MinTTY (the default terminal of MSYS2)**]

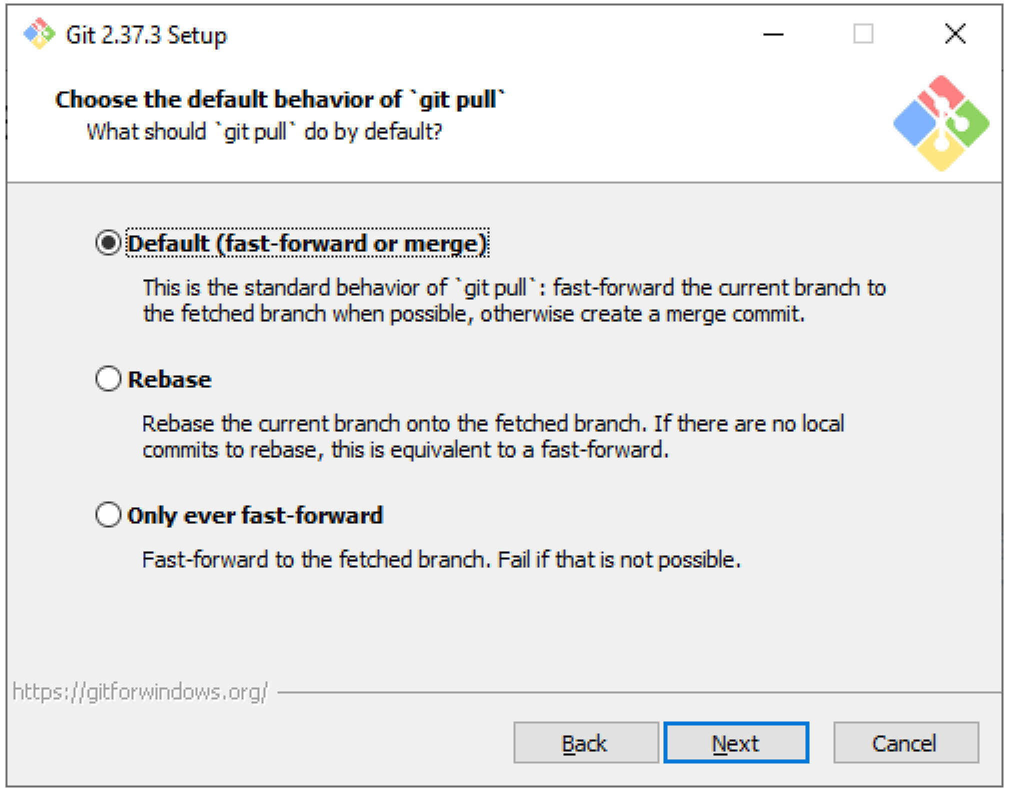
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

Use Windows' default console window

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

Use the default.

Install Git


Git 2.37.3 Setup

Choose the default behavior of `git pull`

What should `git pull` do by default?

[✓ **Default (fast-forward or merge)**]

This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

Rebase

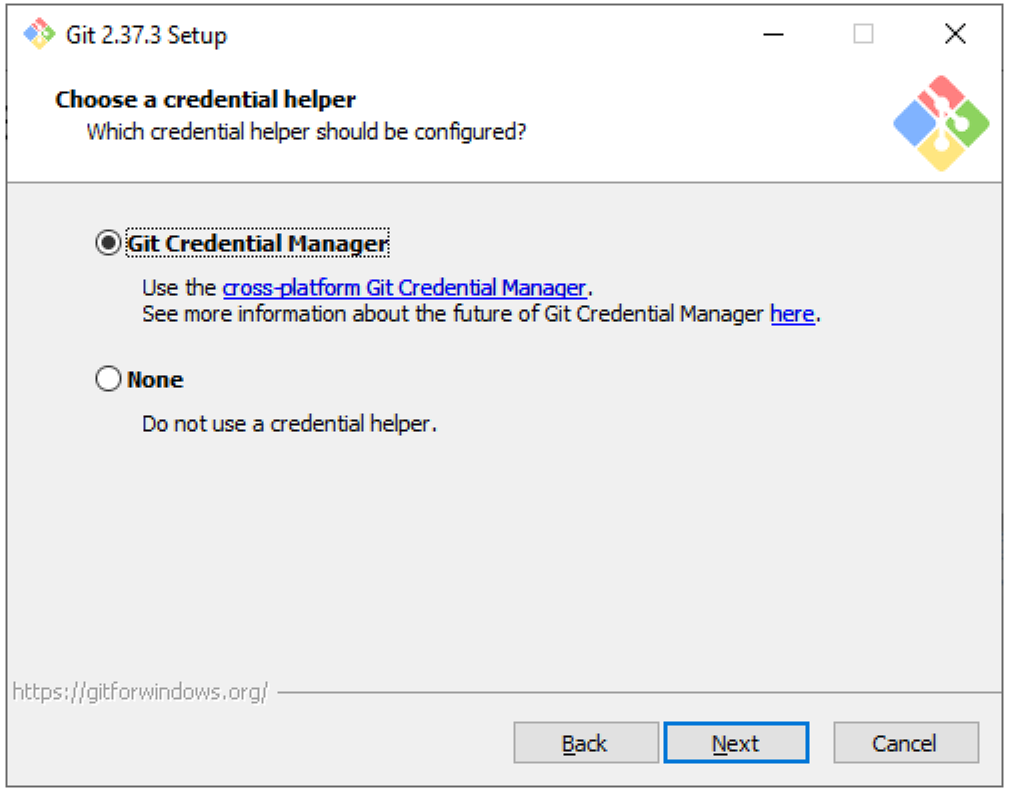
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

Only ever fast-forward

Fast-forward to the fetched branch. Fail if that is not possible.

Use the default.

Install Git


Git 2.37.3 Setup

Choose a credential helper
Which credential helper should be configured?

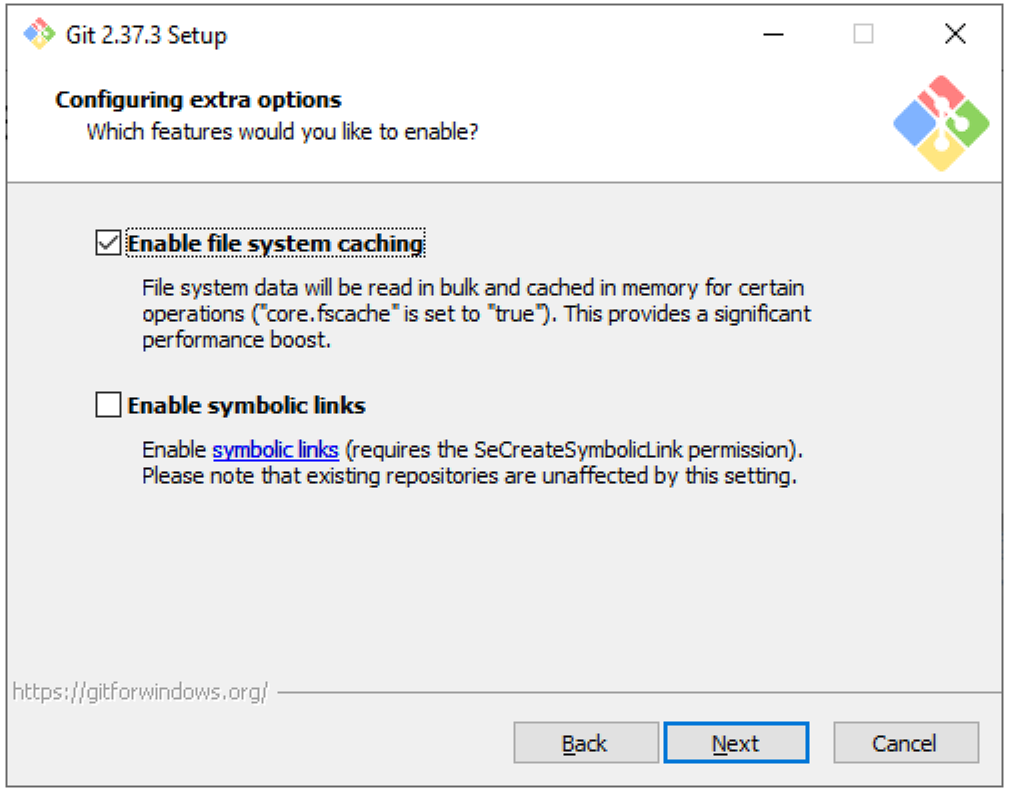
[✓ **Git Credential Manager**]
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.

None
Do not use a credential helper.

https://gitforwindows.org/

Use the default.

Install Git



Git 2.37.3 Setup

Configuring extra options

Which features would you like to enable?

[✓] **Enable file system caching**

File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

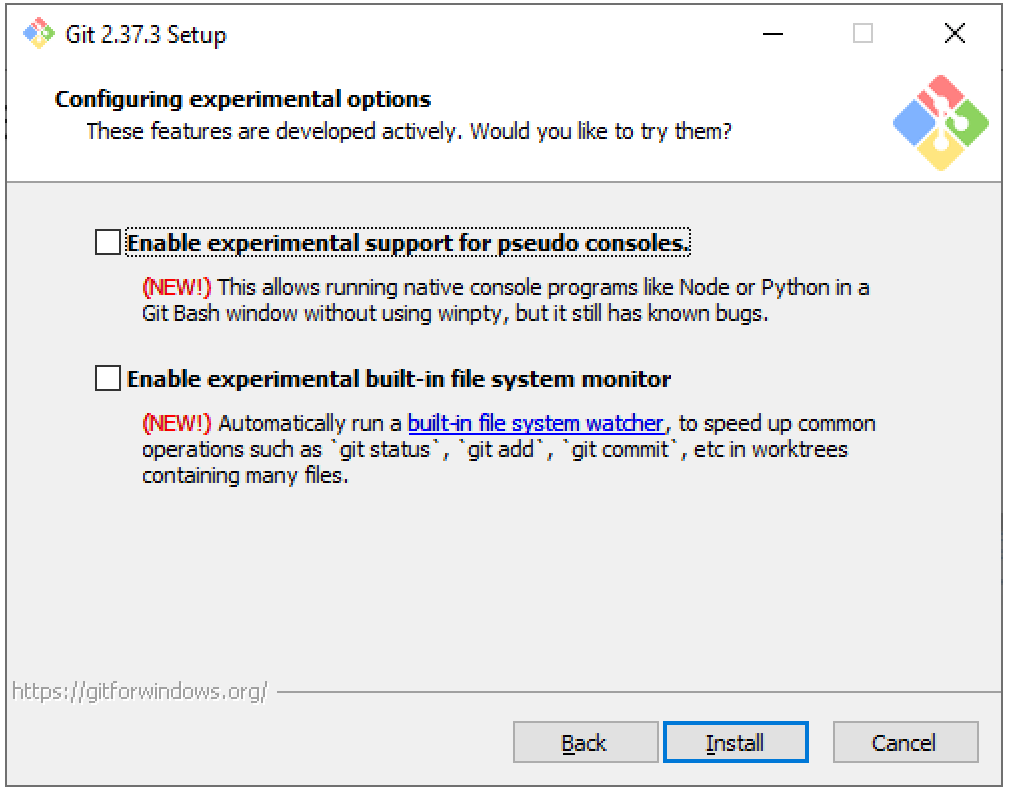
[ ] **Enable symbolic links**

Enable symbolic links (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.

https://gitforwindows.org/

Use the default.

Install Git



Git 2.37.3 Setup

Configuring experimental options

These features are developed actively. Would you like to try them?

☐ **Enable experimental support for pseudo consoles.**

(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.

☐ **Enable experimental built-in file system monitor**

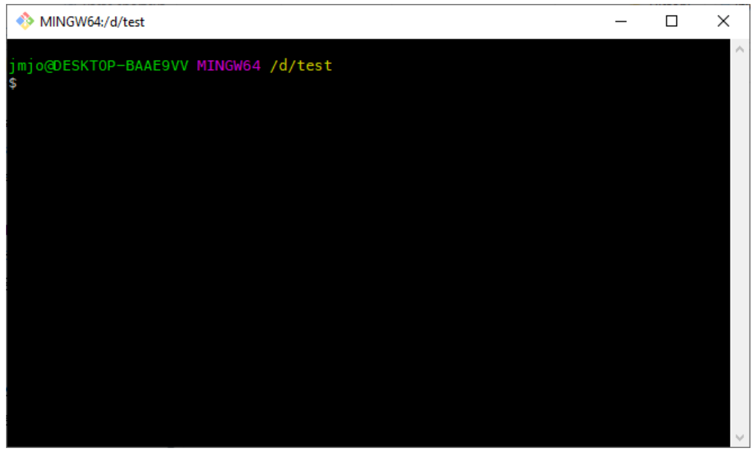
(NEW!) Automatically run a built-in file system watcher, to speed up common operations such as \`git status\`, \`git add\`, \`git commit\`, etc in worktrees containing many files.

https://gitforwindows.org/

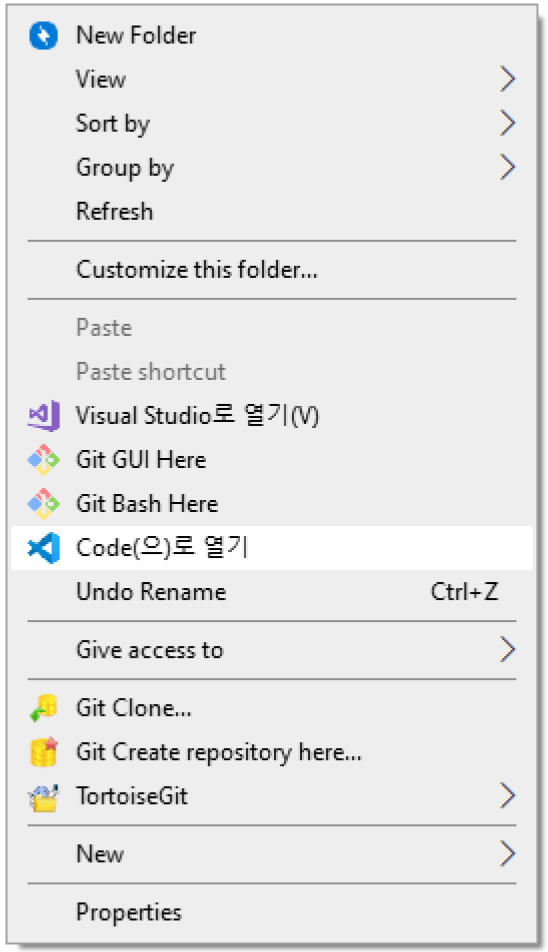
Use the default.

Opening VS Code and Git Bash

1. Go to the directory
2. Open the context menu
3. Open with Code
4. Git Bash Here


MINGW64:/d/test

```
jmjo@DESKTOP-BAAE9VV MINGW64 /d/test
$
```



Let's Practice – 1 (init)

1. Initialize a repo: `git init`
2. Create README.md: `code README.md`
3. Write and save
4. Stage the file: `git add README.md`
5. Make a commit: `git commit`
6. Make sure a commit has been made: `git log`

- Type `git status` between the commands to see the status.

Let’s Practice – 2 (commit)

1. Change README.md: `code README.md`
2. Add and commit at once: `git commit –a`

Let’s Practice – 3 (amend)

1. Change README.md: `code README.md`
2. Add, commit, and leave a message at once: `git commit –a –m <msg>`
3. Want to modify the last commit? `git commit ––amend`

Let’s Practice – 4 (rm)

1. Create *nouse.txt*: >> `nouse.txt`
2. `git add .`
3. `git commit`
4. Let’s remove it: `git rm nouse.txt`
5. Check if the file is removed: `ls`
6. ~~`git add .`~~ (why we use `git rm` instead of just rm)
7. `git commit`

- Type `git status` between the commands to see the status.

Let’s Practice – 5 (restore)

1. Overwrite README.md: `echo “oops” >> README.md`
2. Restore it: `git restore README.md`
3. Check if the content is restored: `cat README.md`

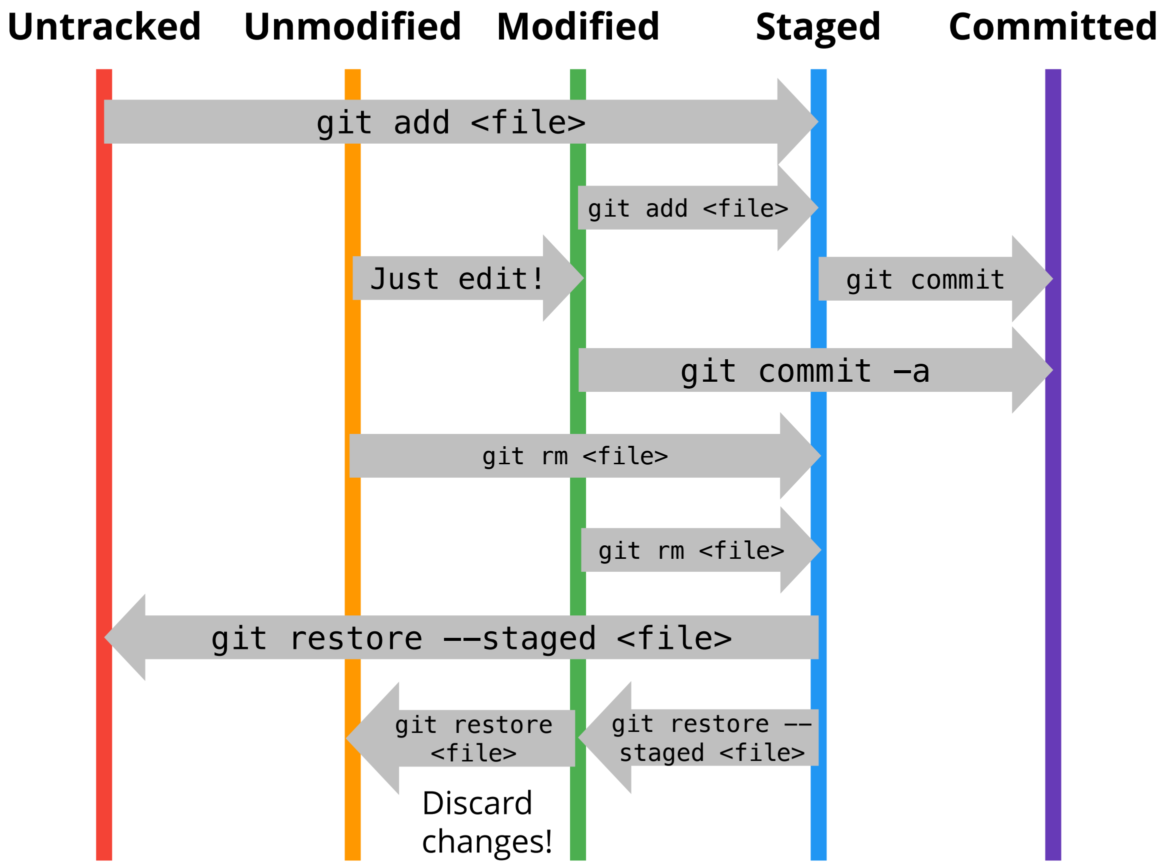
- Type `git status` between the commands to see the status.

Let's Practice – 6 (gitignore)

1. Make some noisy CSV files:
   - `>> 1.csv`
   - `>> 2.csv`
   - `>> 3.csv`
2. Stage them by mistake: `git add .`
3. Let's fix this. Unstage all first: `git restore --staged .`
4. Open .gitignore: `code .gitignore`
5. Write `*.csv` to .gitignore and save.
6. Stage the files again: `git add .`
7. Check the status to see if csv files are ignored: `git status`

Let's Practice – 7

- Freely practice the commands below.



Questions..

- You can work with Git commands on the VS Code interface without using Git Bash. Can you figure it out?

- Can you create an empty directory and add it to repo?
  - Hint: .gitkeep

- What happens if you rename a file? Any difference between `mv` (or via the File Explorer) and `git mv`?

- What does the message below mean and when does it happen?
  - `warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it`